PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-240548
(43)Date of publication of application : 11.09.1998

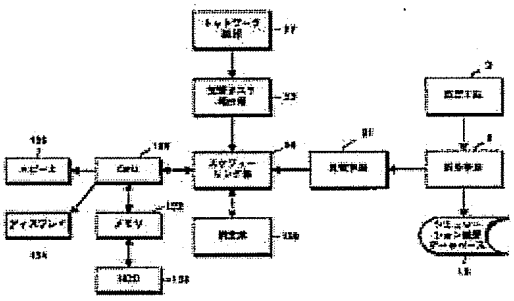
(51)Int.Cl. G06F 9/46

(21)Application number : 09-047990
(22)Date of filing : 03.03.1997

(71)Applicant : TOSHIBA CORP
(72)Inventor : FUJIWARA YASUSHI

(54) TASK SCHEDULING DEVICE AND METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To schedule tasks corresponding to desired service quality while economically utilizing a CPU while using fixed priority scheduling.
SOLUTION: A specifying means 3 specifies the service quality to be provided by the execution of the task and a sectioning means 6 sections the supplied respective tasks into an essential part to be surely executed so as to satisfy the specified service quality and an option part to be executed when possible. A scheduling part 39 schedules the essential part and schedules the option part only in the case that CPU time remains after the execution of the essential part is ended.



(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平10－240548

(43)公開日　平成10年(1998)９月11日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/46	３４０	Ｇ０６Ｆ　9/46	３４０Ｅ

審査請求　未請求　請求項の数８　ＯＬ　（全 16 頁）

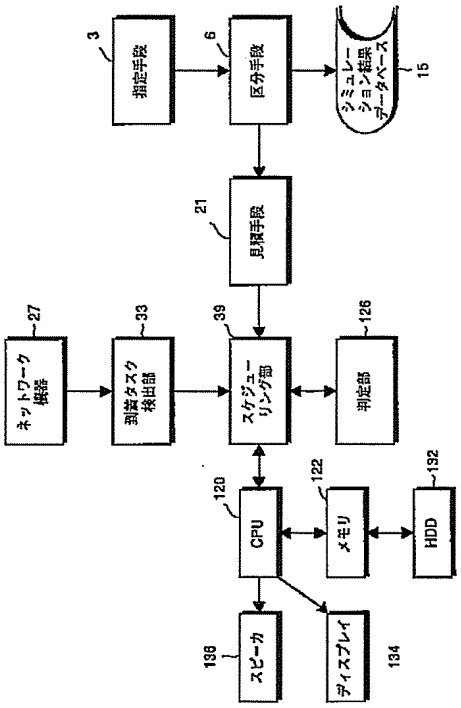
(21)出願番号　特願平9－47990

(22)出願日　平成９年(1997)３月３日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　藤原　靖
神奈川県川崎市幸区柳町70番地　株式会社東芝柳町工場内
(74)代理人　弁理士　木内　光春

(54)【発明の名称】　タスクスケジューリング装置及び方法

(57)【要約】
【課題】　固定優先順位スケジューリングを用いながら、ＣＰＵを経済的に活用しつつ、所望のサービス品質に対応するタスクスケジューリングを実現する。
【解決手段】　指定手段３が、タスクの実行によって提供されるべきサービス品質を指定する。区分手段６が、与えられた各タスクを、指定されたサービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分する。スケジューリング部３９が、必須部分のスケジューリングを行い、必須部分の実行終了後にＣＰＵ時間が余る場合に限って、前記オプション部分をスケジューリングする。



【特許請求の範囲】
【請求項１】　与えられた複数のタスクを、サービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分する区分手段と、各タスクの前記必須部分のスケジューリングと、各タスクの前記オプション部分のスケジューリングを行うスケジューリング手段と、
を有することを特徴とするタスクスケジューリング装置。
【請求項２】　前記スケジューリング手段は、
必須部分の優先順位をオプション部分の優先順位よりも高く設定し、
優先順位の高いタスクにＣＰＵの実行権を順次与え、
全ての必須部分の実行が終了した場合に、オプション部分を実行することを特徴とする請求項１記載のタスクスケジューリング装置。
【請求項３】　区分された前記オプション部分をスケジューリングするためのサブスケジューリング手段を有することを特徴とする請求項１又は２記載のタスクスケジューリング装置。
【請求項４】　区分された必須部分の実行所要時間を見積る見積手段と、
見積られた前記実行所要時間に基づいて、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かを判定する判定手段と、
を有することを特徴とする請求項１，２又は３記載のタスクスケジューリング装置。
【請求項５】　サービス品質と、タスク中の必須部分と、当該必須部分の実行所要時間との対応関係をあらかじめ記憶させたデータベースを有し、
前記区分手段及び前記見積手段は、前記データベースに記憶された対応関係を参照することによって、前記タスクの区分及び前記実行所要時間の見積を行うように構成されたことを特徴とする請求項４記載のタスクスケジューリング装置。
【請求項６】　前記スケジューリング手段及び前記サブスケジューリング手段のうち少なくとも一方は、
所定量のＣＰＵ時間を与えた場合に各タスクの実行によって生じる効用を、あらかじめ所定の基準で計算したデータを記憶させた第２のデータベースを用い、
次に実行するタスクを複数のタスクから選択する場合に、前記複数のタスクのそれぞれにＣＰＵ時間を与えた場合の効用を前記第２のデータベースから参照し、
当該効用が最大のタスクを次に実行するタスクとして選択するように構成されたことを特徴とする請求項１，２，３，４又は５記載のタスクスケジューリング装置。
【請求項７】　与えられた複数のタスクを、サービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分するステップと、各タスクの前記必須部分のスケジューリングと、各タスクの前記オプション部分のスケジューリングを行うステップと、
を含むことを特徴とするタスクスケジューリング方法。
【請求項８】　前記スケジューリングの際に、
所定量のＣＰＵ時間を与えた場合に各タスクの実行によって生じる効用を、あらかじめ所定の基準で計算したデータを記憶させた第２のデータベースを用い、
次に実行するタスクを複数のタスクから選択する場合に、前記複数のタスクのそれぞれにＣＰＵ時間を与えた場合の効用を前記第２のデータベースから参照し、
当該効用が最大のタスクを次に実行するタスクとして選択することを特徴とする請求項７記載のタスクスケジューリング方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、マルチメディア・ネットワーク等の高度な要求に柔軟に対応する組み込みシステム、及びマルチメディアデータ通信システムのための、タスクスケジューリング技術に関わるもので、特に、ＣＰＵを経済的に活用しつつ、所望のサービス品質に対応するタスクスケジューリングを実現するものである。
【０００２】
【従来の技術】コンピュータの急速な普及に伴って、コンピュータを組み込みシステムとして普及する場合が増大している。ここで、組み込みシステムとは、パーソナルコンピュータのようにコンピュータとして独立して使用されるものではなく、家電製品などの電気機器に組み込まれる小規模で機能が制限されたコンピュータシステムである。
【０００３】これまでの組み込みシステムは、冷蔵庫や炊飯器のコントローラなど比較的単純なものが多く、８ビット程度のＣＰＵで十分実現可能であった。しかし最近のマルチメディア・ネットワークの多様な普及につれ、生活環境のなかに存在する組み込みシステムへの要求は高度化しつつある。将来的には家電製品もネットワークに接続され、動画や音声などマルチメディアデータをやり取りすることになると思われる。これらの用途では、３２ビット以上のＣＰＵの使用が当然のこととなると思われる。高速で強力なＣＰＵの使用は、これまでハードウェアで実現してきた機能をソフトウェア化する可能性をもたらすので、ＣＰＵ上で多くのタスクを効率的に実行するためのタスクのスケジューリング技術が一層重要となってきている。
【０００４】特に、動画や音声などマルチメディアを扱うアプリケーションでは、性能の保証が重要である。動画や音声データは、処理のデッドラインが守れない場合、音声が途切れたり、動画がコマ落ちしたりして、品質劣化としてユーザに容易に認識される。このため、これらの分野において、デッドラインを意識したタスクス

ケジューリング、いわゆるリアルタイムスケジューリングが必要である。マルチメディアシステムをこの意味でリアルタイムシステムと考えることも可能だが、マルチメディアシステムに課せられる時間制約は程度問題であって、エレベータ制御における安全管理機能など、これまでの厳密なリアルタイムシステムの時間制約とは、基本的に異なる。これまでのリアルタイムシステムにとって、デッドラインは絶対であった。

【０００５】例えば、戦闘機制御システムでは、外部センサーが一定時間毎にデータをサンプリング・処理する。デッドラインを守れないと、撃墜される可能性があるので、デッドラインを守ることが最重要課題である。このように、デッドラインを守ることが最大の優先課題であるようなシステムを、ハードリアルタイムシステムという。これに対して、マルチメディアシステムでは、ある処理をいつまでに必ず行う、という制約は少なく、各種の再生などを一定の品質で行うことの方が重要である。このように厳しいデッドライン制約を持たないリアルタイムシステムは、しばしばソフトリアルタイムシステムと呼ばれる。

【０００６】タスクスケジューリングは、オペレーティングシステムの基礎であり、リアルタイムシステムに適用可能なスケジューリングとしては、これまでに多くの手法が提案されている。組み込みシステムに関連するリアルタイムスケジューリングについては、軍事・航空管制などを主なターゲットにした方式がいくつか提案されている。

【０００７】複数のタスクをスケジューリングする手法としては、実行可能状態にある各タスクに優先順位（優先度）を割当て、最も高い優先順位を持つタスクに、ＣＰＵ時間（実行権）を各タスクに与える優先順位スケジューリングが一般的である。この方面で代表的な方式としては、周期的に発生するタスクを周期が短いものほど優先順位を高く設定して、その優先順位に応じスケジューリングを行うレートモノトニック方式や、デッドラインまでの残り時間が短いタスクほど優先順位を高く設定し、その優先順位に応じスケジューリングを行うＥＤＦ（Ｅａｒｌｉｅｔ　Ｄｅａｄｌｉｎｅ　Ｆｉｒｓｔ）方式などがある（“Ｓｃｈｅｄｕｌｉｎｇａｌｇｏｒｉｔｈｍｓ　ｆｏｒ　ｍｕｌｔｉ－ｐｒｏｇｒａｍｍｉｎｇ　ｉｎａ　ｈａｒｄ　ｒｅａｌ－ｔｉｍｅ　ｅｎｖｉｒｏｎｍｅｎｔ”ｂｙ Ｃ．Ｌ．Ｌｉｕ ａｎｄ Ｊ．Ｗ．Ｌｅｙｌａｎｄ，ｉｎ Ｊｏｕｒｎａｌ ｏｆ ｔｈｅ Ａｓｓｏｃｉａｔｉｏｎ ｆｏｒ Ｃｏｍｐｕｔｉｎｇ Ｍａｃｈｉｎｅｒｙ ２０，１（Ｊａｎｕａｒｙ １９７３）：４０－６１”）。

【０００８】このように、優先順位に基づくスケジューリング（優先順位スケジューリング）は、優先順位が固定である固定優先順位スケジューリングと、優先順位が動的に変化する動的優先順位スケジューリングに大別され、レートモノトニック方式は固定優先順位スケジューリングの、ＥＤＦ方式は動的優先順位スケジューリングの、それぞれ代表的な例となっている。

【０００９】固定優先順位スケジューリングでは、タスクに割り当てられた優先順位は、タスクの生成から消滅まで不変である。レートモノトニックスケジューリングは、周期的に実行が必要となるタスクのためのスケジューリング方式で、周期が短いタスクほど優先度を高く設定するものである。例えば、画像データが５０ｍｓｅｃに一度到達し、音声データ図３０ｍｓｅｃに一度到達するとしよう。この時レートモノトニックスケジューリングでは、周期のより短い音声データを音声化するタスクの優先順位を、画像データを画像化するタスクの優先順位よりも高く設定する。

【００１０】図９は、優先順位スケジューリングの一実行例である。ここでは、三つの周期的タスクτ１，τ２，τ３があり、それぞれが周期１５ｍｓ，２５ｍｓ，４０ｍｓで、周期的に起動される。また、一度の周期で、各タスクは７ｍｓだけ実行する。タスクの優先順位は、タスクτ１が一番高く、タスクτ２が二番目、タスクτ３の優先順位が最も低い、と仮定する。この優先順位は、レートモノトニック方式によるものである。

【００１１】この場合、時点０で、全てのタスクが起動されるが、優先順位の一番高いタスクτ１にＣＰＵでの実行権が与えられ、７ｍｓ経過した後、最初の周期を終了する。その後直ちに、タスクτ２にＣＰＵでの実行権が与えられ、７ｍｓ経過した１４ｍｓの時点でその最初の周期を終了する。その後、タスクτ３にＣＰＵの実行権が渡されるが、１５ｍｓ時点でタスクτ１がその二度目の周期を開始するため、ＣＰＵでの実行権はタスクτ３からタスクτ１へと移される。２２ｍｓ時点までタスクτ１は実行し、そこで二度目の周期を終える。２２ｍｓ時点では、タスクτ２の二度目の周期はまだ始まっていないので、ＣＰＵはタスクτ３に与えられる。しかし、２５ｍｓ時点でタスクτ２が二度目の周期を開始するため、タスクτ３は３ｍｓだけ実行した後、タスクτ２に実行権を譲り渡す。

【００１２】このタスクτ２も３０ｍｓ時点でタスクτ１が三度目の周期を開始するため、５ｍｓだけ実行した後、タスクτ１にＣＰＵを譲り渡す。３７ｍｓ時点で、タスクτ１はその三度目の周期を終え、優先順位が二番目であるタスクτ２にＣＰＵの実行権が与えられる。タスクτ２は残っていた２ｍｓの仕事を行い、３９ｍｓ時点でその二度目の周期を終える。３９ｍｓでタスクτ３に実行権が戻ってきたが、その最初の周期が終わる４０ｍｓまでには、残された７－１－３＝３ｍｓ分の実行は行えず、タスクτ３の二度目の周期が開始されてしまい、４０ｍｓの時点でこのデッドラインを破ってしまう。もっとも、優先順位のいちばん高いタスクτ１の実行は確保されており、このことがこの種のスケジューリ

ングの特徴である。その後も、このように優先順位に基づいてタスクがスケジューリングされていく。

【００１３】一般に、固定優先順位スケジューリングの方が、実現の容易さ・過負荷時の挙動の予測可能性などから選好される傾向があり、リアルタイムスケジューリング理論ではレートモノトニック方式が主流になっている。マルチメディア関係でも、動画や音声は周期的に発生・到着することから、ネットワークに接続されたマルチメディアファイルサーバに対して、レートモノトニック方式を階層的ラウンドロビン方式と併せ適用しようという提案（ＵＳＰ５５２８５１３）などがある。

【００１４】

【発明が解決しようとする課題】しかし、従来提唱されているリアルタイムスケジューリング技術は、マルチメディアアプリケーション特有の要求を十分満たしてはいない。まず、マルチメディア対応の組み込みシステムでは、コストの問題から、ＣＰＵの能力にはあまり余裕がないのが普通である。しかし、既存のリアルタイムスケジューリング手法は、デッドラインを守るためＣＰＵの能力にはかなり余裕を持たせるのが通常であり、経済的コストの面で不利益を生じる。このため、ＣＰＵを効率よく利用するタスクスケジューリング装置が求められていた。

【００１５】また、マルチメディアアプリケーションに特徴的なものとして、サービス品質（Ｑｕａｌｉｔｙ　ｏｆ　Ｓｅｒｖｉｃｅ，ＱｏＳ）の保証がある。ユーザがマルチメディアアプリケーションにデータの再生要求を出す際は、再生の質についての要求も行う。これをサービス品質の指定というが、マルチメディアアプリケーションは、サービス品質の指定に応じたスケジューリングを行う必要がある（“Ｒｅｓｏｕｒｃｅ　ｍａｎａｇｅｍｅｎｔ　ｉｎ　ｎｅｔｗｏｒｋｅｄ　ｍｕｌｔｉｍｅｄｉａ　ｓｙｓｔｅｍｓ”ｂｙ　Ｋ．Ｎａｈｒｓｔｅｄｔ　ａｎｄＲ．Ｓｔｅｉｎｍｅｔｚ　ｉｎ　ＩＥＥＥ　Ｃｏｍｐｕｔｅｒ　２８，５（Ｍａｙ　１９９５）：５２－６３）。

【００１６】これまでのリアルタイムスケジューリング技術では、サービス品質の指定への対応は考えられていなかった。すなわち、レートモノトニック方式では、タスクの優先順位はその周期だけで決まってしまい、ユーザによるサービス品質指定を反映できない。

【００１７】一方、優先順位が状況に応じて変化する動的優先順位スケジューリングでは、固定優先順位スケジューリングと比べ柔軟なスケジューリングが可能であり、サービス品質の指定への対応も可能と思われ、サービス品質が重要なマルチメディアシステムでは、動的優先順位に基づくスケジューリング手段（スケジューラ）の使用も提案されている。しかし、動的優先順位スケジューリングには、実現コストが大きい・過負荷時の挙動が予測できないといった課題が残されている。

【００１８】優先順位スケジューリングではないが、要求に柔軟に対応するスケジューリング手法としては、ｉｍｐｒｅｃｉｓｅ　ｃｏｍｐｕｔａｔｉｏｎと呼ばれるものも存在する（“Ｉｍｐｒｅｃｉｓｅ　ｃｏｍｐｕｔａｔｉｏｎｓ”ｂｙ　Ｊ．Ｗ．Ｓ．Ｌｉｕ　ｅｔ　ａｌ．ｉｎ　Ｐｒｏｃｅｅｄｉｎｇｓ　ｏｆ　ｔｈｅ　ＩＥＥＥ８２，１（Ｊａｎｕａｒｙ　１９９４）：８３－９４）。この手法は、それぞれのタスクを実行が必須な部分とオプショナルの部分に分割し、必須部分の実行が終了した後オプショナルの部分を実行する。オプショナルの部分は、最後まで実行することなく途中で実行を中止することが可能である。ｉｍｐｒｅｃｉｓｅ　ｃｏｍｐｕｔａｔｉｏｎでは投入時間とサービス品質はトレードオフの関係にあり、実行時間を長くするほど得られるサービス品質は向上するが、利用できる時間には制限がある。この仮定は、動画再生など多くのアプリケーションに当てはまる。

【００１９】しかし、タスクが複数個存在した場合、ｉｍｐｒｅｃｉｓｅ　ｃｏｍｐｕｔａｔｉｏｎではオプショナルな部分をスケジュールするためには、非線形計画法やマルコフ決定過程など大掛かりな数学的道具を必要とする。また、ｉｍｐｒｅｃｉｓｅ　ｃｏｍｐｕｔａｔｉｏｎでは、スケジューリングを行うのに要する時間が、タスクの個数について指数的に増加するので、組み込みシステムのようにシステム自体は小規模だが、多くのタスクが存在する場合への適用は困難である。

【００２０】本発明は、上記のような従来技術の問題点を解決するために提案されたもので、その目的は、優先順位スケジューリングを用いながら、ＣＰＵを経済的に活用しつつ、所望のサービス品質に対応するタスクスケジューリングを実現することである。また、本発明の他の目的は、従来のタスクスケジューリング装置を用いたシステムへの導入が容易なタスクスケジューリング装置を提供することである。

【００２１】

【課題を解決するための手段】上記の目的を達成するため、請求項１のタスクスケジューリング装置は、与えられた複数のタスクを、サービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分する区分手段と、各タスクの前記必須部分のスケジューリングと、各タスクの前記オプション部分のスケジューリングを行うスケジューリング手段と、を有することを特徴とする。請求項７のタスクスケジューリング方法は、請求項１の発明を方法の観点から把握したものであって、与えられた複数のタスクを、サービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分するステップと、各タスクの前記必須部分のスケジューリングと、各タスクの前記オプション部分のスケジューリングを行うステップと、を含むことを特徴とする。

【００２２】請求項１，７の発明では、サービス品質を指定することができ、タスクは、指定されたサービス品質を満足するのに必須な必須部分と、それ以外のオプション部分に区分される。そして、まず、必須部分のスケジューリングを、レートモノトニック方式のような従来の固定優先順位によって行う。レートモノトニック方式のような従来のリアルタイムスケジューリング理論を用いた部分は厳密なスケジューリング可能性解析が可能であり、このようなスケジューリングを必須部分に適用することによって、サービス品質が確実に保証できる。オプション部分は、必須部分の実行が終了してなおかつ、ＣＰＵ時間が余る場合に限って実行する。このため、マルチメディア・ネットワークなどにおいて負荷が過剰の場合も、特に指定されたサービス品質に係る必須部分は他の部分より優先実行されるので、サービス品質が保証される。必須部分以外のオプション部分もＣＰＵ時間に余裕がある場合は実行されるので、ＣＰＵが経済的・効率的に有効活用され、サービス品質が一層向上する。

【００２３】請求項２の発明は、請求項１記載のタスクスケジューリング装置において、前記スケジューリング手段は、必須部分の優先順位をオプション部分の優先順位よりも高く設定し、優先順位の高いタスクにＣＰＵの実行権を順次与え、全ての必須部分の実行が終了した場合に、オプション部分を実行することを特徴とする。

【００２４】請求項２の発明では、優先順位を可変としておき、必須部分の優先順位をオプション部分よりも高くしておく。この状態で、固定優先順位のスケジューリングを行うことにより、必須部分のみが先に実行される。必須部分の実行が終了した場合は、今度はオプション部分が実行対象となる。このようにすれば、必須部分とオプション部分に、一貫して固定優先順位スケジューリングを適用できる。このため、固定優先順位スケジューリングを用いた従来の組み込みシステムに、本発明を適用しようとする場合も、最小限の追加・変更で済む。

【００２５】請求項３の発明は、請求項１又は２記載のタスクスケジューリング装置において、区分された前記オプション部分をスケジューリングするためのサブスケジューリング手段を有することを特徴とする。請求項３の発明では、オプション部分のスケジューリングを専用のサブスケジューラに任せるので、必須部分をスケジューリングする主なスケジューラの負担が軽減され、より経済的なスケジューリングが可能である。

【００２６】請求項４の発明は、請求項１，２又は３記載のタスクスケジューリング装置において、区分された必須部分の実行所要時間を見積る見積手段と、見積られた前記実行所要時間に基づいて、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かを判定する判定手段と、を有することを特徴とする。請求項４の発明では、必須部分の実行所要時間が見積られ、さらに、見積られた実行所要時間を用いて、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かが判定される。すなわち、指定されたサービス品質を前提として、スケジューリング可能かどうかがシステムの稼働前に判断できるので、ＣＰＵの負担過剰によるシステム障害を事前に回避できる。なお、見積られた実行所要時間は、指定したサービス品質が実現できるかどうかや、オプション部分がどのくらい実行できそうかをユーザなどが判断しようとする場合に、ＣＰＵ時間の余裕を容易に判断するために用いてもよい。

【００２７】請求項５の発明は、請求項４記載のタスクスケジューリング装置において、サービス品質と、タスク中の必須部分と、当該必須部分の実行所要時間との対応関係をあらかじめ記憶させたデータベースを有し、前記区分手段及び前記見積手段は、前記データベースに記憶された対応関係を参照することによって、前記タスクの区分及び前記実行所要時間の見積を行うように構成されたことを特徴とする。請求項５の発明では、あらかじめデータベースに記憶させた情報を用いて、サービス品質から必須部分を定めたり、必須部分の実行所要時間を得ることができるので、処理が正確かつ迅速に行われる。

【００２８】請求項６の発明は、請求項１，２，３，４又は５記載のタスクスケジューリング装置において、前記スケジューリング手段及び前記サブスケジューリング手段のうち少なくとも一方は、所定量のＣＰＵ時間を与えた場合に各タスクの実行によって生じる効用を、あらかじめ所定の基準で計算したデータを記憶させた第２のデータベースを用い、次に実行するタスクを複数のタスクから選択する場合に、前記複数のタスクのそれぞれにＣＰＵ時間を与えた場合の効用を前記第２のデータベースから参照し、当該効用が最大のタスクを次に実行するタスクとして選択するように構成されたことを特徴とする。請求項８の発明は、請求項６の発明を方法の観点から把握したものであって、請求項７記載のタスクスケジューリング方法において、前記スケジューリングの際に、所定量のＣＰＵ時間を与えた場合に各タスクの実行によって生じる効用を、あらかじめ所定の基準で計算したデータを記憶させた第２のデータベースを用い、次に実行するタスクを複数のタスクから選択する場合に、前記複数のタスクのそれぞれにＣＰＵ時間を与えた場合の効用を前記第２のデータベースから参照し、当該効用が最大のタスクを次に実行するタスクとして選択することを特徴とする。請求項６，８の発明では、ＣＰＵ時間を与えた場合の効用（効果）が最大のタスクが実行されるので、システムの性能が最大限に発揮される。

【００２９】

【発明の実施の形態】以下、本発明の複数の実施形態について、図面を参照して詳細に説明する。

【００３０】１．第１実施形態

（１）構成

図１は、第１実施形態の全体構成を示すブロック図であり、具体的には、本発明によるスケジューリング装置とネットワーク及び周辺装置、典型的にはインターネット端末やビデオオンデマンドなどのマルチメディア機器である。ここでは説明を具体的にするため、ネットワークに接続されたマルチメディアクライアント・サーバシステムを題材にとり、図１のシステムは、マルチメディアサーバ１０５、複数のマルチメディアクライアント１０７（請求項１にいうタスクスケジューリング装置に相当するもの）、さらにそれらの間のネットワーク１０３などからなるものとする。

【００３１】マルチメディアサーバ１０５は、ＣＰＵ１１０とメモリ１１２、タスクのスケジューラ１１４、スケジューリング可能かどうかの判定部１１６、大規模なＤＶＤや光ディスクドライブなどからなる記憶装置（ストレージ）１０１を持ち、ネットワーク１０３で接続されている。

【００３２】マルチメディアクライアント１０７も、ＣＰＵ１２０とメモリ１２２、タスクのスケジューラ１２４、スケジューリング可能かどうかの判定部１２６を持つ。さらに、クライアント１０７は、マルチメディアデータの再生・表示用に、ディスプレー１３４・スピーカ１３６などのＡＶ機器に接続されている。なお、マルチメディアクライアント１０７も、ハードディスクドライブ（ＨＤＤ）など二次の補助記憶１３２を有し、マルチメディアサーバ１０５からのデータを格納する場合がある。マルチメディアクライアント１０７は、ネットワーク１０３を経由して、マルチメディアサーバ１０５とつながっている。

【００３３】なお、ここでは、動画データ・音声データなどのメディアデータの記録・再生を、コンピュータによって制御・統合するシステムをマルチメディアシステムと呼んでいる。さらにここでは、マルチメディアシステムはスタンドアローンではなく、ネットワークによって結合されていると想定している。このようなシステムにおいて、動画データ・音声データなどの送受信やそれに付随したデータの圧縮・伸長などが典型的なマルチメディアタスクである。サーバーからは毎秒３０フレーム送り出される動画データなどがこれにあたる。この場合の周期は３３ｍｓで、一つのフレームをどの程度のサービス品質で再生するかによって、ＣＰＵ消費時間は異なる。

【００３４】このようなデータを処理するマルチメディアアプリケーションでは、サーバからクライアントまでの間で、サービス品質（ＱｏＳ：Ｑｕａｌｉｔｙ　ｏｆ　Ｓｅｒｖｉｃｅ）の保証が要求される。ＱｏＳの保証には、サーバやクライアント以外にも、ネットワークノードの役割が大きいが、ここでは、主にサーバ・クライアントにおけるスケジューリングについて説明する。

【００３５】次に、図２は、クライアント１０７の具体的な構成を示す機能ブロック図であり、スケジューラ１２４の構成要素を具体的にし、ＣＰＵ１２０など、他の要素と共に示したものである。すなわち、クライアント１０７は、請求項１のタスクスケジューリング装置に対応するもので、タスクの実行によって提供されるべきサービス品質を指定するための指定手段３を有する。また、クライアント１０７は、与えられた各タスクを、指定されたサービス品質を満足するために必ず実行すべき必須部分と、可能ならば実行すべきオプション部分に区分する区分手段６と、この区分に用いるデータを格納しているシミュレーション結果データベース１５（請求項５にいうデータベースに相当するもの）と、を有する。

【００３６】また、クライアント１０７は、区分された必須部分の実行所要時間を見積る見積手段２１（請求項４）と、必須部分のスケジューリングを行い、必須部分の実行終了後にＣＰＵ時間が余る場合に限って、前記オプション部分をスケジューリングするスケジューリング部３９（請求項１にいうスケジューリング手段に相当するもの）と、を有する。また、クライアント１０７は、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かを判定する判定部１２６（請求項４にいう判定手段に相当するもの）を有する。

【００３７】また、クライアント１０７は、ネットワーク機器２７から到着するタスクを検出する到着タスク検出部３３と、スケジューリング部３９によるスケジューリングにしたがって、タスクを実行するＣＰＵ１２０と、ＣＰＵ１２０とともに情報処理や情報出力を行うためのメモリ１２２、ハードディスクユニット（ＨＤＤ）１３２、スピーカ１３６及びディスプレイ１３４を有する。

【００３８】（２）作用及び効果

上記のような構成を有する第１実施形態は、次のような作用を有する。すなわち、マルチメディアクライアント１０７からは、マルチメディアサーバ１０５に対し、データの送信・停止・再送などの要求が送られる。マルチメディアサーバ１０５は、マルチメディアクライアント１０７からの要求により、マルチメディアデータを記憶装置１０１から検索し、ネットワーク１０３を通じてクライアント１０７に送信する。マルチメディアサーバ１０５からのデータの送信は、多くの場合、データを処理するタスクの周期的な実行を必要とする。

【００３９】マルチメディアサーバ１０５は、多数のマルチメディアクライアント１０７からの要求に対応する必要がある。これらのマルチメディアクライアント１０７の要求はユーザのサービス品質指定を反映したサービス品質指定を持つので、マルチメディアサーバ１０５でも、サービス品質保証を目的としたタスクスケジューリングが必要である。またマルチメディアクライアント１０７は、ひとつのマルチメディアサーバ１０５から複数

のデータストリームを受信する場合も、複数のマルチメディアサーバー１０５から同時にデータを受信する場合もある。マルチメディアクライアント１０７のスケジューラは、それぞれのデータストリームに対し、ユーザが指定したサービス品質が提供できるよう、タスクスケジューリングを行う。このように、マルチメディアサーバ１０５・マルチメディアクライアント１０７は、共にサービス品質を保証するためのスケジューリングを必要とするが、両者で共通のスケジューリング手法が適用できること、構成が簡単であること、本実施形態での例が組み込みシステムであることから、ここではマルチメディアクライアント１０７のＣＰＵでのタスクスケジューリングについて説明する。

【００４０】すなわち、マルチメディアクライアント１０７のＣＰＵのスケジューラ１２４に求められるのは、サービス品質指定を充足しつつ過負荷時にも破局的な結果をもたらさないよう、マルチメディアアプリケーションをスケジューリングすることである。スケジューラ１２４の構成は、レートモノトニックスケジューリングをベースにしたものである。

【００４１】（２－１）サービス品質の指定

まず、指定手段３により、サービス品質の指定が行われる。サービス品質の指定は、ユーザがその都度行ってもよいが、タスクやデータの種類によってデフォルト値が自動的に選択されるようにしてもよい。ユーザによるサービス品質指定は、以下により、本スケジューリング装置に反映される。なお、サービス品質の例は、動画における１秒間のフレーム数や、音声におけるサンプリング周波数などである。多くのタスクでは割り当てられたＣＰＵ時間によって、サービス品質が定まると考えられる。ユーザの指定したサービス品質を保証するために必要なＣＰＵ使用時間を、タスクの中での必須部分の消費するＣＰＵ使用時間として設定する。

【００４２】（２－２）タスクの区分

サービス品質が指定されると、区分手段６が、タスクを、必須部分とオプション部分に区分する。この区分は、例えば、次のように行うことができる。すなわち、シミュレーション等を行うことによって、タスク中どこまで実行すればどの程度のサービス品質が得られるかのデータを収集してデータベース化し、要求されるサービス品質と必須部分やその実行所要時間との対応関係を判定しておく。この場合、必須部分の実行所要時間は、シミュレーションで求めるだけでなく、プログラムの実行時間を測定する従来の装置を用いて得ることができる。そして、必須部分の量と実行所要時間との対応関係もデータベースに保存しておく。

【００４３】このような情報は、シミュレーション結果データベース１５（図２）に記憶させておく。ユーザがサービス品質を指定すると、区分手段６がデータベース１５を照会してどの部分を必須部分に区分するかを決定する。通常は、必須部分はタスクの最初の部分であり、オプション部分はタスクのその後の部分である。さらに、見積手段２１が、必須部分ごとの実行所要時間を前記データベースから読み出して実行時間を見積ることによって、サービス品質を充足するために必要なＣＰＵ時間を計算する。

【００４４】なお、判定部１２６は、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かを判定し、可能と判定された場合に次のスケジューリングが行われる（請求項４）。このように、第１実施形態では、指定されたサービス品質を前提として、スケジューリング可能かどうかがシステムの稼働前に判断できるので、ＣＰＵの負担過剰によるシステム障害を事前に回避できる。

【００４５】（２－３）スケジューリング

続いて、スケジューリング部３９が、各タスクの必須部分について、レートモノトニック方式に従ってスケジューリングを行う。レートモノトニックスケジューリングは、スケジューリング可能性の厳密な判定が可能なので、必ず実行されるべき部分が実際スケジューリング可能であること、言い換えれば指定されたサービス品質が提供できることを厳密に保証できる。レートモノトニック方式では、スケジューリング部３９は、必須部分の優先順位をオプション部分の優先順位よりも高く設定し、優先順位の高いタスクにＣＰＵの実行権を順次与え、全ての必須部分の実行が終了した場合に、オプション部分を実行する（請求項２）。

【００４６】すなわち、スケジューリング部３９は、まず、必須部分の優先順位を高く設定し、オプション部分の優先順位を低く設定する。ここで、図３は、必須部分とオプション部分に区分されたタスクを示す概念図である。ここでは、３つのタスクτ1 、τ2 、τ3 が存在し、タスクτ1 は必須部分τ′1及びオプション部分τ”1に区分され、タスクτ2 は必須部分τ′2及びオプション部分τ”2に区分され、タスクτ3 は必須部分を含まずにオプション部分τ”3のみから構成されるものとする。この場合、実行対象は、図３に示すように、優先順位が高い必須部分３１０と優先順位が低いオプション部分３２０の二つに分かれる。必須部分３１０は具体的には、必須部分τ′１（３１２）及びτ′２（３１４）で、ユーザのサービス品質指定を保証するために必ず実行する必要のある部分である。オプション部分３２０は具体的には、オプション部分τ”1（３２２）、τ”2（３２４）及びτ”3（３２６）で、必須部分必須部分τ′１（３１２）及びτ′２（３１４）の実行が終了して余裕がある場合に限り実行対象となる。

【００４７】タスクτ1 （３１２），τ2 （３１２）が、ユーザ品質指定を保証するに十分なだけ実行していない間は、必須部分τ′１（３１２），と必須部分τ′２（３１２）を対象としてレートモノトニック方式でス

ケジューリングが行われる。この時点では、矢印３１６及び３１８のように、必須部分τ′１（３１２），と必須部分τ′２　（３１２）との間でタスクの切り替えが行われる。しかし、タスクτ１，τ2 について、ユーザ品質指定を保証するに十分なだけ必須部分τ′ 1及びτ′ 2を実行したら、その後、実行対象タスクの切り替えが矢印３３１，３３２のように行われ、オプション部分τ"1、τ"2、τ"3が、高い優先順位で実行されるタスクがない間だけ、実行される。

【００４８】図４は、本実施形態におけるスケジューリングの処理の流れを表したフロー図である。すなわち、スケジューリング部３９は、現在実行可能なタスクのうち、最も高い優先順位の与えられたタスクを選びそのタスクにＣＰＵの実行権を与える（ステップ４２０）。タスクはＣＰＵ１２０で実行されるが（ステップ４３０）、その間にネットワーク機器２７から新しいタスクが届いた場合は（ステップ４４０）、その到着を到着タスク検出部３３が検出し、その新しく到着したタスクも含めて優先順位最高のものを調べ直す（ステップ４２０）。また、ＣＰＵで実行していたタスクが終了したら（ステップ４５０）、残りのタスクの中で優先順位最高のものを選び直す（ステップ４２０）。また、高い優先順位で実行される時間が終了したら（ステップ４６０）、必須部分の優先順位を下げた後（ステップ４７０）、どのタスクにＣＰＵを与えるか決定するために優先順位を調べ直す（ステップ４２０）。

【００４９】例えば、優先順位として、当初、必須部分τ′ 1に１、必須部分τ′ 2に２、オプション部分τ"1に３、オプション部分τ"2に４、オプション部分τ"3に５を設定したとする。この場合、当初は、優先順位が高い必須部分τ′ 1又は必須部分τ′ 2のうち、待ち状態ではなく実行可能状態にあるものが交互に実行される。もちろん、必須部分τ′ 1及びτ′ 2の双方が待ち状態の時は、オプション部分τ"1などを実行してよい。必須部分のための時間帯が終了すると、必須部分τ′ 1及びτ′ 2はもはやＣＰＵを要求しないので、実行対象がオプション部分τ"1、τ"2、τ"3に移る。なお、他の態様として、必須部分τ′ 1及びτ′ 2の優先順位をそれぞれ６と７に引き下げることによって優先順位を逆転させる手法も考えられる。このようにすると、オプション部分τ"1、τ"2、τ"3の優先順位の方が、必須部分τ′ 1及びτ′ 2の優先順位よりも高くなるので、実行対象がオプション部分τ"1、τ"2、τ"3に移る。

【００５０】（２－４）スケジューリングに適した条件

なお、第１実施形態によるスケジューリングに適した条件について説明する。各マルチメディアタスクτi （１≦ｉ≦ｎ）が周期的に発生するとし、その周期をＴｉで表わす。ユーザのサービス品質指定を満足するために最低限実行する必要がある部分を実行するのに必要なＣＰＵ使用時間を、Ｃｉ とする。ここで、Ｃｉ ≦Ｔｉ を仮定するのは当然である。なぜなら、もしＣｉ ＞Ｔｉ ならば、タスクの次の発生までのＣＰＵ時間を全てこのタスクに割り当てても、ユーザが指定するサービス品質は得られず、ユーザのサービス品質指定に達しないからである。タスクの新しいインスタンスが発生した場合は、古いインスタンスを破棄することも当然であり、我々はスケジューラにこの性質を仮定する。

【００５１】タスクτi の、ユーザのサービス品質指定を満足するため実行する必要がある部分を必須部分τ′ iで、それ以外の部分をオプション部分τ"iで表わす。以下では説明を簡単にするため、必須部分τ′ ｉ，オプション部分τ"iをタスクであるかのように議論する。以下の議論では、タスク自体よりその消費時間を問題にしているので、必須部分τ′ ｉ，オプション部分τ"iを導入することで議論が簡単になるからである。

【００５２】次の量をユーザの指定するサービス品質に関するＣＰＵ消費要求と呼び、μで表わす。

【数１】

$$\mu = \Sigma_{i=1}^{n} C_i/T_i$$

この値は、個々のタスクが要求するＣＰＵ占有割合の合計であり、個々のタスクのＣＰＵ消費要求は、指定されたサービス品質を確保するために必要なＣＰＵ使用時間を、タスクの発生周期で除したものである。μ＞１の場合は、ユーザのサービス品質指定が実現が不可能であることが、直ちに分かる。さらに、前掲Ｌｉｕ－Ｌａｙｌａｎｄの論文でも結果として示されているように、次の関係が成り立つ場合は、各タスクτi の一部分必須部分τ′ iを、それぞれ周期Ｔi のタスクとみなし、周期の短いものほど優先順位を高く設定して、優先順位に従ってスケジューリングを行うと、全ての必須部分τ′ iは次の周期までに実行される。

【数２】

$$\mu \leq n(2^{1/n} - 1) \sim \log 2 = 0.697 (n \rightarrow \infty)$$

これは、全てのタスクのＣＰＵ使用率の合計が約７０％までであれば、すべてのタスクτi について、指定のサービス品質が提供される、すなわち、サービス品質指定が必ず実現可能であることを意味する。すなわち、タスク数ｎを無限大にすると、μは０．６９７に収束するが、第１実施形態ではμを０．６９７以下とする。これ以上にした場合でも特定の条件の場合には、スケジューリング可能であるが、

【数３】

$$n(2^{1/n} - 1) \leq \mu \leq 1$$

の場合は、組み合わせ的な性質により実現可能か否かが微妙に影響されるため、スケジューリング可能であるための必要十分条件を数値的に与えることはできない。この点、安全サイドにたって、ユーザ品質指定の実現可能性保証を与えるのは、

【数４】

$$\mu \leq n(2^{1/n}-1)$$

の場合に限ることが適当と考えられる。一方で、μを上記の範囲以外にする場合を含め、実行が必要な部分がレートモノトニック方式でスケジューリング可能かどうかを判定するための必要十分条件も知られているので、それを利用することも可能である。

【００５３】これまで述べてきたのは必須部分τ′ iが一つ以上実行可能状態な場合であり、実行可能な必須部分τ′ iが存在しないとき、オプション部分τ"i達をどうスケジューリングするか、については、何も述べていない。このような場合のスケジューリング方式は多数考えられるが、ここでは必須部分τ′ iの実行が終了した時点で、オプション部分τ"iの優先順位を、必須部分τ′ｉ（１≦ｉ≦ｎ）で最大のものよりもさらに低いものに設定する。そのうえで、スケジューリングは、これまで通りの優先順位スケジューリングに基づいて行われる。この方式の利点は、これまでのシステムをそのままでサービス品質制御に転用できることである。この記述から明らかなように、固定優先順位スケジューリング（フラットスケジューリング方式）を応用した第１実施形態は、従来技術とは異なった優れた効果を有するものである。特に、第１実施形態は、従来の固定優先順位スケジューリングに、タスクの優先順位変更機能を追加することによって、わずかの変更で実現できる点で優れている。

【００５４】一方で問題となるのは、どのように優先度を変更したらよいかである。組み込みシステムでは、コスト性能比が重要であり、必須部分τ′ iがＣＰＵを使用していないときでも、できるだけ有効にＣＰＵ資源を利用することが望ましい。

【００５５】なお、フラットスケジューリング方式の性能は、優先順位決定アルゴリズムに依存する側面があるので、単にやみくもに優先度を下げるだけでは、ユーザにとり好ましいサービス品質を提供することは困難である。一例として、タスク数をｎとして、必須部分を終えたら優先順位をｎ＋１に設定する、などの単純な方法に限定することは、サービス品質の視点からは必ずしも良い選択ではない。優先順位を決定するアルゴリズムとして種々考えられるが、本来の優先度を考慮したり、タスクの発生周期を考慮するなど、さまざまな手法がありうる。選択肢の例として、タスク数をｎ、必須部分に設定された優先度がｉであったとき、必須部分の実行を終えたら優先順位を

・ｎ＋１に設定する
・ｎ＋ｉに設定する
・２ｎ＋１－ｉに設定する

などが考えられるが、他の様々なバリエーションも検討すべきである。そして、サービス品質の支点からどれがよい選択であるかは、具体的な適用対象や適用目的に応じて、実験やシミュレーションなどを行い、具体的に決定すべきである。

【００５６】（２－５）第１実施形態の効果

以上説明したように、第１実施形態によれば、固定優先順位スケジューリング手法と優先順位調整機能を利用することで、既存のシステムに対する変更を最小限にとどめつつ、これまでの組み込みシステムでは困難だったマルチメディア・ネットワークアプリケーションにおけるサービス品質保証が可能になる。特に、第１実施形態におけるスケジューリング方式は、リアルタイムスケジューリング理論に根拠を置いており、厳密なサービス品質を保証できる。

【００５７】すなわち、第１実施形態では、サービス品質が指定される。タスクは、サービス品質の満足に必須な必須部分と、それ以外のオプション部分に区分される。そして、まず、必須部分のスケジューリングを、レートモノトニック方式のような従来の固定優先順位によって行う。レートモノトニック方式のような従来のリアルタイムスケジューリング理論を用いた部分は厳密なスケジューリング可能性解析が可能であり、このようなスケジューリングを必須部分に適用することによって、サービス品質が確実に保証できる（請求項１，７）。オプション部分は、必須部分の実行が終了してなおかつ、ＣＰＵ時間が余る場合に限って実行する。このため、マルチメディア・ネットワークなどにおいて負荷が過剰の場合も、特に指定されたサービス品質に係る必須部分は他の部分より優先実行されるので、サービス品質が保証される。必須部分以外のオプション部分もＣＰＵ時間に余裕がある場合は実行されるので、ＣＰＵが経済的・効率的に有効活用され、サービス品質が一層向上する。

【００５８】また、第１実施形態におけるスケジューリングでは、優先順位を可変としておき、必須部分の優先順位をオプション部分よりも高くしておく。この状態で、固定優先順位のスケジューリングを行うことにより、必須部分のみが先に実行される。必須部分の実行が終了した場合は、今度はオプション部分が実行対象となる。このようにすれば、必須部分とオプション部分に、一貫して固定優先順位スケジューリングを適用できる。このため、固定優先順位スケジューリングを用いた従来の組み込みシステムに、本発明を適用しようとする場合も、最小限の追加・変更で済む（請求項２）。

【００５９】また、第１実施形態では、必須部分の実行所要時間が見積られ、さらに、見積られた実行所要時間を用いて、指定されたサービス品質を満足するように各タスクがスケジューリング可能か否かが判定される。すなわち、指定されたサービス品質を前提として、スケジューリング可能かどうかがシステムの稼働前に判断できるので、ＣＰＵの負担過剰によるシステム障害を事前に回避できる（請求項４）。なお、見積られた実行所要時

間は、指定したサービス品質が実現できるかどうかや、オプション部分がどのくらい実行できそうかをユーザなどが判断しようとする場合に、ＣＰＵ時間の余裕を容易に判断するために用いてもよい。

【００６０】また、第１実施形態では、あらかじめデータベース１５に記憶させた情報を用いて、サービス品質から必須部分を定めたり、必須部分の実行所要時間を得ることができるので、処理が正確かつ迅速に行われる（請求項５）。

【００６１】特に、第１実施形態は通信ネットワークに接続されているので、通信ネットワークを経由して画像や音声などのマルチメディアデータが複数種類送信される場合も、画像や音声などに応じた複数のタスクをスケジューリングすることによって、指定されたサービス品質を保証することができる。

【００６２】２．第２実施形態

（１）構成

第２実施形態のタスクスケジューリング装置は、第１実施形態と略同様の構成を有するが、次の点で第１実施形態と異なるので、第１実施形態との相違点を中心に説明する。すなわち、図５は、第２実施形態のタスクスケジューリング装置の構成を示す機能ブロック図である。この図に示すように、第２実施形態は、第１実施形態と異なり、区分された前記オプション部分をスケジューリングするためのサブスケジューリング部８７（請求項３にいうサブスケジューリング手段に相当するもの）と、サブスケジューリング部８７がタスクを選択するためのデータを記録したタスク効用データベース９３（請求項６，８にいう第２のデータベースに相当するもの）と、を有する。

【００６３】（２）作用及び効果

第１実施形態においては、オプション部分τ"iは優先順位を低く設定し、後に必須部分とオプション部分の優先順位を逆転させたが、スケジューラ自体は、一貫して優先順位に従い、優先順位の高い各必須部分τ′ jと同様の手法で、オプション部分τ"iをスケジューリングしていた。このため、オプション部分τ"iにどのような優先順位を与えるのが適当かが問題となる。

【００６４】第２実施形態では、各オプション部分τ"iをもっぱらスケジューリングする機構を導入する。第２実施形態では、各オプション部分τ"iは、これまでのレートモノトニック方式によるスケジューラの監督を離れ、異なるスケジューラの監督下に入る。新たなスケジューラとその監督下にある各オプション部分τ"iは、本来のスケジューラからはひとつのタスクであるかのように見えるという意味で、このスケジューリング方式は階層的である。

【００６５】この新たなスケジューラ（サブスケジューリング部８７）を、ここではサブスケジューラと呼ぶ。それに対し、いままで考えてきた本来のスケジューラ（スケジューリング部３９）を、主スケジューラと呼ぶ。サブスケジューラが複数であることも可能であるし、サブスケジューラ自体がさらに階層的な構造を持つ可能性もある。ここでは二階層からなる階層的スケジューリングを述べるが、本発明は三階層以上の階層的スケジューリングに適用することもできる。

【００６６】階層的スケジューリング方式の目的は、サービス品質の一層の向上とＣＰＵ資源の有効利用である。ユーザの指定した最低限のサービス品質を保証しただけでは、最大の性能を発揮してユーザの期待に応えることはできない。ＣＰＵに余裕がある限り、より高いサービス品質が提供されることがユーザにとって望ましい。ユーザが指定した以上のサービス品質に対応する部分をいかにスケジューリングするかも重要な問題になる。ここでは、全体としてのサービス品質を向上させることを目的として、マルチメディアアプリケーションをスケジューリングする階層的スケジューリング方式の一実現方法を述べる。

【００６７】図６は、第２実施形態におけるタスクの構成を示す概念図である。この図に示すように、必須部分τ′１（６０２），必須部分τ′２（６０４）は、ユーザの品質指定を満足するために割り当てられた時間をまだ使いきっていない間、主スケジューラによってレートモノトニック方式による優先順位が割り当てられ、スケジューリングが行われている。図３に示した第１実施形態の場合と異なるのは、ユーザの品質指定を満足するために割り当てられた時間を使いきった後、オプション部分τ"1（６１２），オプション部分τ"2（６１４），オプション部分τ"3（６１６）は、主スケジューラによるスケジューリングの対象とはならず、これら３つ合わせたオプション部分群６１０として、一括してサブスケジューラによりスケジューリングされることである。オプション部分τ"1，オプション部分τ"2，オプション部分τ"3は、主スケジューラからは、サブスケジューラによってまとめてスケジューリングされるため、ひとつのタスクのように見え、この意味で第２実施形態におけるスケジューリングは階層的である。

【００６８】すなわち、必須部分τ′１（６０２）及びτ′２（６０４）間での制御移転（６０６，６０８）は主スケジューラによって行われるが、オプション部分τ"1（６１２），τ"2（６１４）及びτ"3（６１６）への制御移転（６２１，６２２）はサブスケジューラが行う。

【００６９】図７は、第２実施形態におけるタスクスケジューリングの処理手順を示すフローチャートである。第２実施形態におけるスケジューリングでは、まず、主スケジューラのタスクキューが空かどうか、すなわちユーザの品質指定を満足するために割り当てられた時間をまだ使いきっていない実行可能タスクがあるかどうかを調べる（ステップ７２０）。もしタスクキューが空な

ら、サブスケジューラに制御が移る（ステップ７５２／後述）。

【００７０】もしタスクキューが空でなければ、その中で最も優先順位の高いものにＣＰＵでの実行権を与える（ステップ７２５）。実行権を与えられたタスクはＣＰＵ上で実行されるが（ステップ７３０）、新しく到着したタスクがあれば（ステップ７３５）、どのタスクにＣＰＵを与えるかを決定するために優先順位を再び比較する（ステップ７２５）。新しく到着したタスクがない場合でも、現在のタスクが終了したなら（ステップ７４０）、主スケジューラのタスクキューが空であるかのチェックに戻る（ステップ７２０）。

【００７１】現在のタスクが終了しない場合でも、もし現在のタスクに割り当てられた高い優先順位で実行する時間が終了したら（ステップ７４５）、現在のタスクを主スケジューラのタスクキューから外しサブスケジューラのタスクキューに移し（ステップ７５０）、その後主スケジューラのタスクキューが空かどうかのチェックに戻る（ステップ７２０）。これらの条件がどれも成立しない限りは、このタスクをＣＰＵ上で実行し続ける（ステップ７３０）。

【００７２】ステップ７２０において、主スケジューラのタスクキューが空の場合、制御はサブスケジューラに移る。この場合、まず、サブスケジューラのタスクキューが空かどうかを調べる（ステップ７５２）、もし空でない場合は、サブスケジューラは所定の基準に基づいて、タスクキューからＣＰＵを割り当てるタスクを選択する（ステップ７５５）。選択されたタスクはＣＰＵ上で実行されるが（ステップ７６０）、その間に新しいタスクが到着した場合は（ステップ７６５）、ＣＰＵを取り上げられて、制御は主スケジューラに移され、主スケジューラのタスクキューをチェックする（ステップ７２０）。

【００７３】新しいタスクが到着せず、現在のタスクが終了したら（ステップ７７０）、サブスケジューラのタスクキューをチェックし（ステップ６５２）、サブスケジューラの基準にしたがって、どのタスク（オプション部分）何をスケジューリングするかを決定する（ステップ７５５）。そうでない場合、主スケジューラからサブスケジューラに移されたタスクがあれば（ステップ７７５）、再びスケジューラのポリシに従いどのタスクにＣＰＵを与えるかを選択する（ステップ７５５）。

【００７４】そうでない場合、クオンタを使い果たしたら（ステップ７８０）、このタスクをＣＰＵから外しサブスケジューラの基準に基づいてどのタスクをスケジューリングするかを決定する（ステップ７５５）。このどれも成立しない限り、タスクはＣＰＵ上で実行され続ける（ステップ７６０）。ステップ７５２において、サブスケジューラのタスクキューが空である場合は（ステップ７５３）、新しいタスクが到着したかどうかを調べ（ステップ７８５）、もし到着すれば主スケジューラに制御を移し（ステップ７２５）、到着しない間はこのチェック（ステップ７８５）を繰り返す。

【００７５】なお、サブスケジューラ８７は、ＣＰＵで実行するタスク（オプション部分）を、次のような基準で選択すればよい。まず、全てのサービス品質は何等かの形で定量化が可能で、異なるタスクのサービス品質が比較できる、との仮定を置く。例えば、動画についてのサービス品質と音声についてのサービス品質が比較できるとする。もちろんこのような数量化は絶対的なものではありえないが、経済学ではしばしば用いられ、その比較可能な量は効用（ｕｔｉｌｉｔｙ）と呼ばれているので、ここでもこの用語法を採用する。ユーザの指定するサービス品質は、それぞれのタスクに関する、効用のしきい値と考えられ、それ以下の効用はユーザに受け入れられない。

【００７６】マルチメディアアプリケーションでは、ＣＰＵ使用時間の割り当てを増加させても、サービス品質はそれほど向上しないことがしばしばある。例えば、ＣＰＵ使用時間を倍にしても、音声再生のサービス品質は倍にならない。本明細書にいう効用に対しても、経済学で言うところの収穫逓減現象が当てはまると思われる。

【００７７】収穫逓減現象は、ＣＰＵ消費時間に対してサービス品質を描いたグラフは、上に凸である、と言い換えられる。この性質は必ずしも成立する必要はないが、以下に述べるタスクの選択基準は、このような場合に最適なものとなっている。また、ＣＰＵ消費時間と効用の関係は既知であり、データベースもしくは計算式等として、利用可能であると仮定する。

【００７８】すなわち、サブスケジューラは、現在実行可能な各オプション部分τ"iそれぞれに対し、次のクオンタ（固定の時間幅）を割り当てたときの効用の増加量を計算し、増加量が最大になるオプション部分τ"iにＣＰＵを割り当てる。なお、このような効用の増加量はあらかじめ計算しておき、その情報をタスク効用データベース９３に記憶させておく。

【００７９】ここで、図８を参照して実例を説明する。説明を単純にするため、二つのタスクτ"1とτ"2があったとし、それぞれの効用と時間との関係が、図８（ａ）のグラフ８１０及び図８（ｂ）のグラフ８３０で与えられているとする。クオンタをδとし、効用の変化量を見ると、タスクτ"1における変化量８２０の方がタスクτ"2における変化量８４０よりも大きい。このため、最初のクオンタはタスクτ"1に割り当てられる。しかし、次のクオンタについては、タスクτ"1に割り当てた場合の効用の変化量８２５よりもタスクτ"2に割り当てた場合の変化量８４０の方が大きいので、次のクオンタはタスクτ"2に割り当てられる。さらにその次のクオンタについても、同様にタスクτ"2に割り当てられる。このような手法によって、ＣＰＵ時間を最も効果的に活用する

ことができる。

【００８０】この方式は、効用に関して常に最適であるとは限らないが、効用がＣＰＵ消費時間の増加に対し非減少かつ上に凸との性質を満たす場合には最適である。もちろん、この方式によるスケジューリング結果は、効用の与え方に大きく依存する。例えば、ＣＰＵ消費時間に対して定数であるとして、効用を与えることも可能であり、この場合はどのようにスケジュールしても良い。

【００８１】ここでは、サブスケジューラの用いるアルゴリズムとして、効用に基づくものを述べたが、それ以外のアルゴリズムを採用することも当然に可能である。例えば、タイムシェアリングシステムでよく用いられるラウンドロビン方式は、タスクに一定のクオンタを割り当て、タスクがそのクオンタを使い果たすと、タスクキューの最後に移され、次のタスクがＣＰＵを割り当てられる、という手法であるが、サブスケジューラの方式として適用できる。

【００８２】以上のように、第２実施形態では、オプション部分のスケジューリングを専用のサブスケジューラに任せるので、必須部分をスケジューリングする主なスケジューラの負担が軽減され、より経済的なスケジューリングが可能になるとともに、一層のサービス品質向上が可能になる。また、第２実施形態では、ＣＰＵ時間を与えた場合の効用（効果）が最大のタスクが実行されるので、システムの性能が最大限に発揮される（請求項６，８）。

【００８３】３．他の実施形態

なお、本発明は上記実施形態に限定されるものではないので、次に例示するような他の実施形態をも包含するものである。例えば、上記各実施形態ではもっぱら組み込みシステムを例に議論したが、この手法はマルチメディア通信システムのスケジューリングにも等しく適用できることは明らかである。

【００８４】また、タスク管理には次のような手法を適用することができる。例えば、区分された前記オプション部分ごとに生存期間を設定する。タスクのオプション部分がＣＰＵに割り当てられようとする場合には、生存期間が所定の値を超えているか否かを判定する。この結果、生存期間が所定の値を超えているタスクは廃棄する。このように、割り付けられそうになったタスクの生存期間をチェックし、タスクが生存期間を越えていればそれを廃棄することで、タスクの生成と消滅が繰り返されるような場合でも、未処理のタスクが増大し続けないようにできる。

【００８５】また、タスクの一部分が選択された場合に、当該選択された部分によって、指定されたサービス品質が満足できるか否かを判定する手段を設けてもよい。このようにすれば、スケジューリングの一部を手作業で行ったり、タスクスケジューリング装置のデバッグを行うことが容易になる。なお、このような判定は、例えば、サービス品質とその実現に必要なタスクの部分との対応関係をあらかじめ調べてデータとして記憶させておき、記憶されているタスクの部分が、選択された部分に含まれているかどうかを調べることによって、実現することができる。

【００８６】なお、本発明のタスクスケジューリング装置及びタスクスケジューリング方法は、コンピュータプログラムを用いて実現されることが一般的と考えられるが、そのようなコンピュータプログラムを記録したフロッピーディスクなどの記録媒体も本発明の一態様である。

【００８７】

【発明の効果】以上説明したように、本発明によれば、ＣＰＵを経済的に活用しつつ、所望のサービス品質に対応するタスクスケジューリングが実現される。

【図面の簡単な説明】

【図１】本発明の第１実施形態におけるマルチメディアクライアント・サーバシステム全体の概略適構成を示すブロック図。

【図２】本発明の第１実施形態におけるクライアントの具体的構成を示す機能ブロック図。

【図３】本発明の第１実施形態におけるタスクを示す概念図。

【図４】本発明の第１実施形態におけるタスクスケジューリングの処理手順を示すフローチャート。

【図５】本発明の第２実施形態におけるクライアントの具体的構成を示す機能ブロック図。

【図６】本発明の第２実施形態におけるタスクを示す概念図。

【図７】本発明の第２実施形態におけるタスクスケジューリングの処理手順を示すフローチャート。

【図８】本発明の第２実施形態における投入時間（クオンタ）とそれによる効用との関係を示すグラフ。

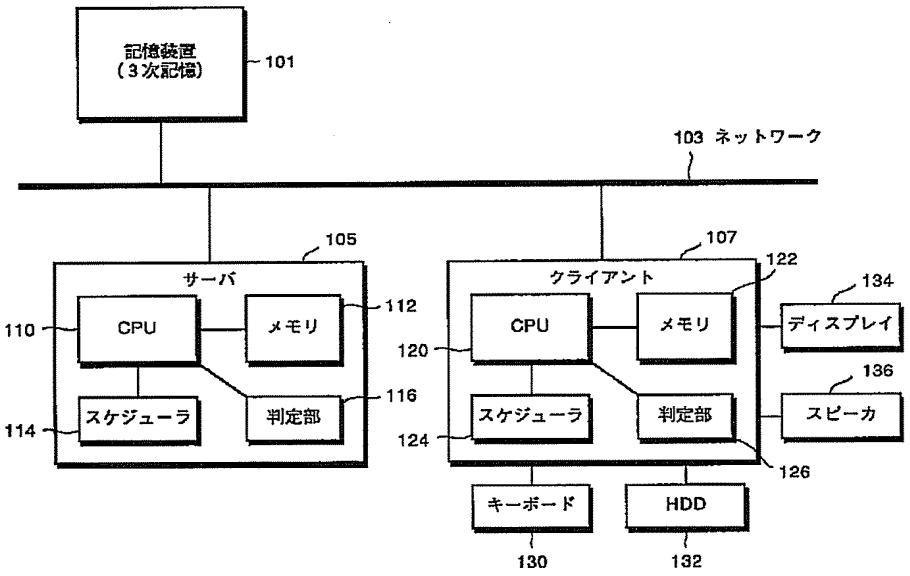
【図９】レートモノトニックスケジューリングの例を示すタイムチャート。

【符号の説明】

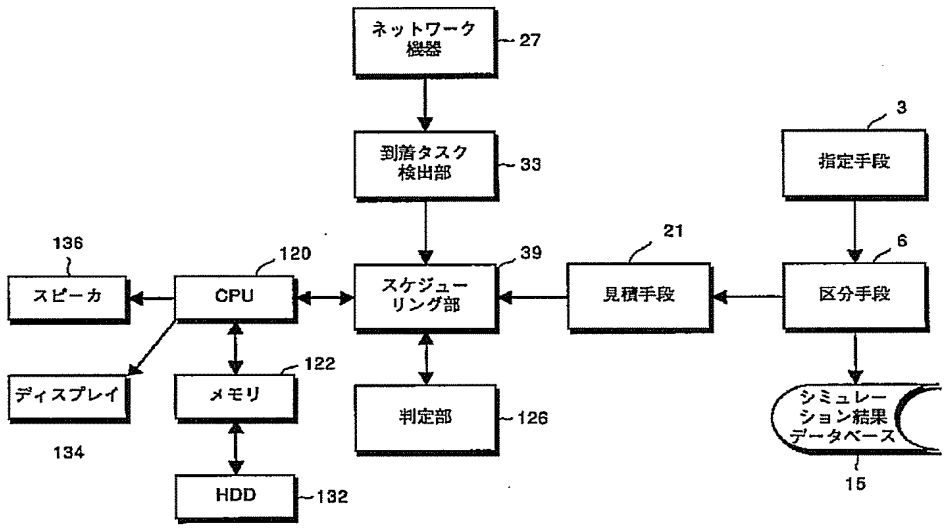
３…指定手段
６…区分手段
１５，９３…データベース
２１…見積手段
２７…ネットワーク機器
３３…到着タスク検出部
３９…スケジューリング部
８７…サブスケジューリング部
１０１…記憶装置
１０３…ネットワーク
１０５…サーバ
１０７…クライアント
１１０，１２０…ＣＰＵ
１１２，１２２…メモリ

１１４，１２４…スケジューラ
１１６，１２６…判定部
１３０…キーボード
１３２…ハードディスクドライブ
１３４…ディスプレイ
１３６…スピーカ
３１０，６０２，６０３…必須部分
３２０，６１０…オプション部分
８１０，８３０…グラフ
８２０，８２５，９４０，９４５…変化量
ＳＴＥＰ…手順の各ステップ

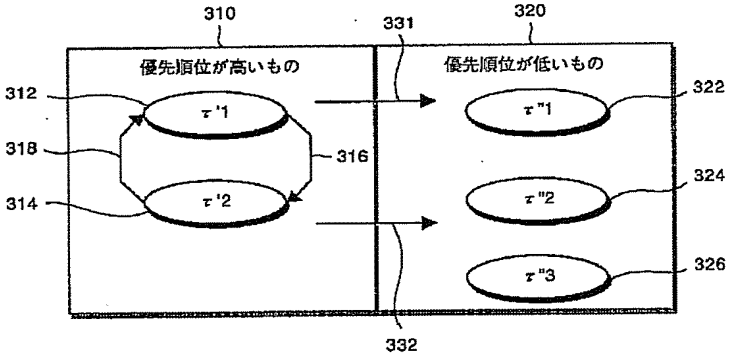
【図１】



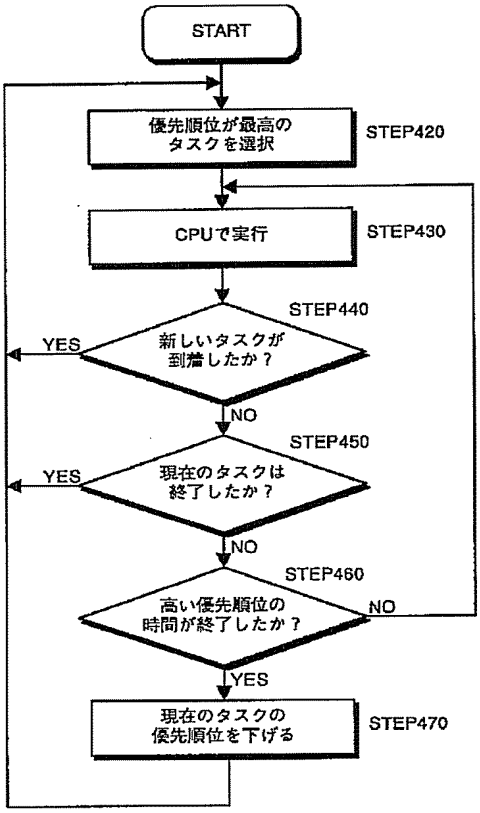
【図２】



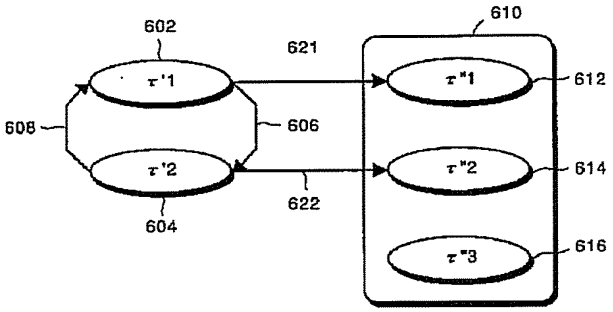
【図３】


310
331
320
優先順位が高いもの
優先順位が低いもの
312
318
314
τ'1
τ'2
316
τ"1
τ"2
τ"3
322
324
326
332

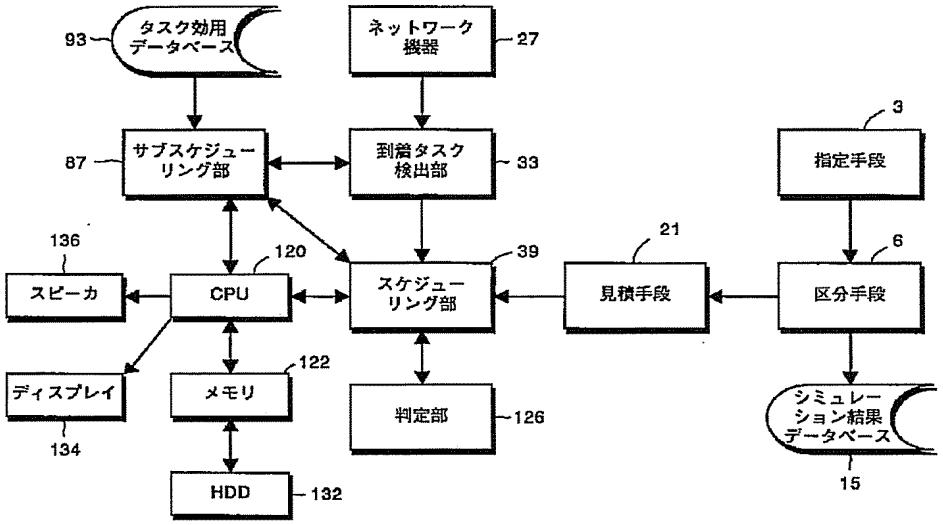
【図４】


START
優先順位が最高の
タスクを選択
STEP420
CPUで実行
STEP430
STEP440
新しいタスクが
到着したか？
YES
NO
STEP450
現在のタスクは
終了したか？
YES
NO
STEP460
高い優先順位の
時間が終了したか？
NO
YES
現在のタスクの
優先順位を下げる
STEP470

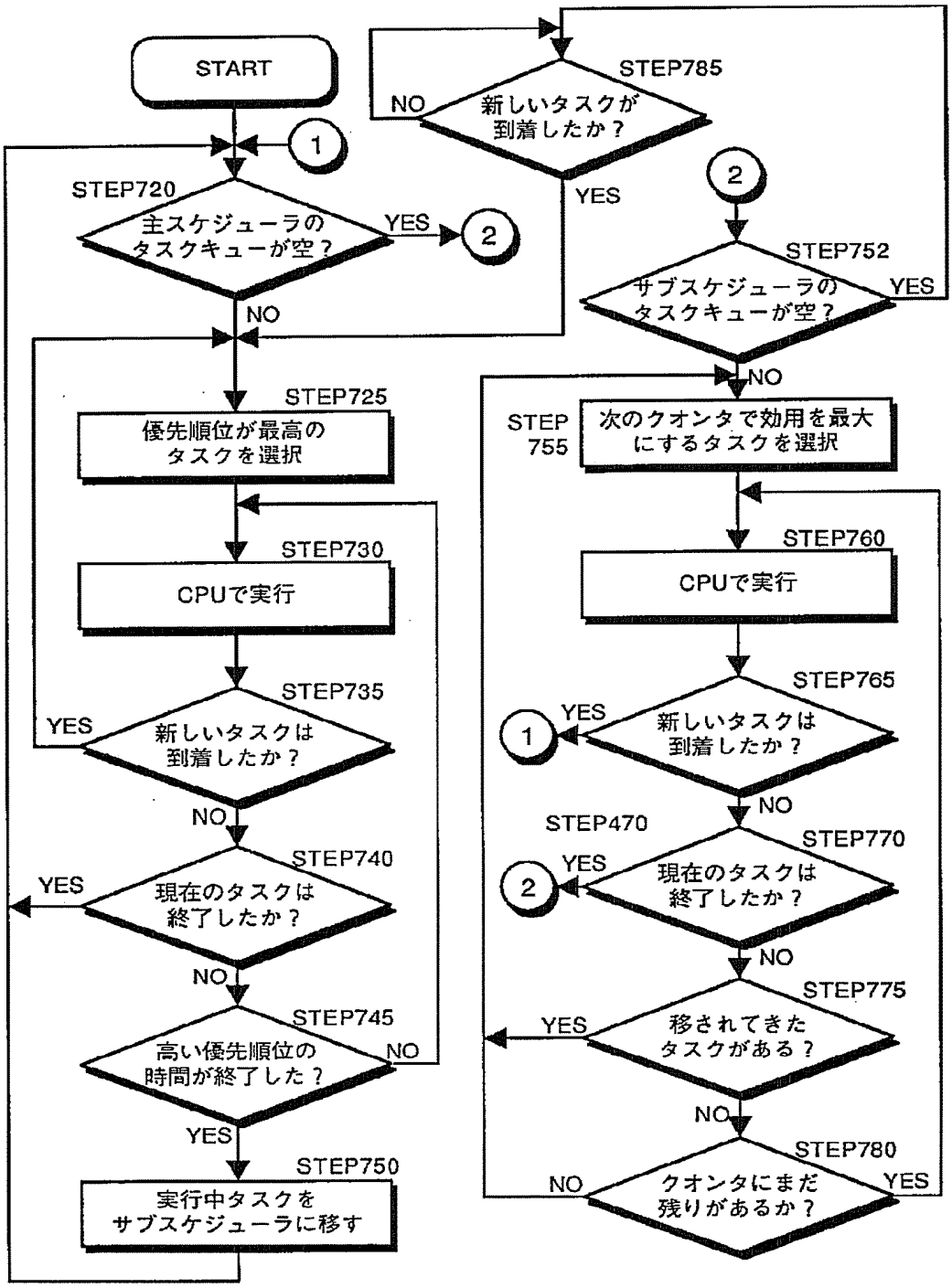
【図６】


602
610
621
τ'1
τ'2
608
606
604
622
τ"1
τ"2
τ"3
612
614
616

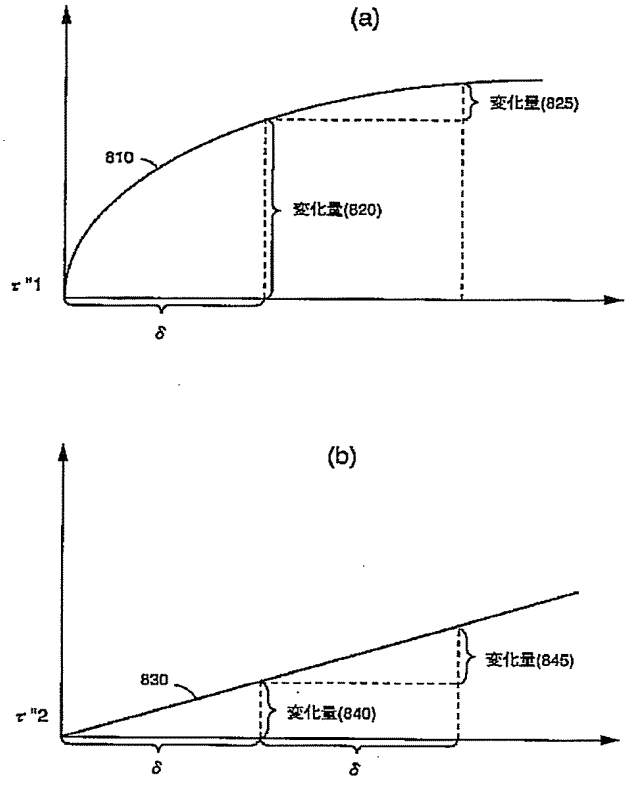
【図５】


93
タスク効用
データベース
ネットワーク
機器
27
87
サブスケジュー
リング部
到着タスク
検出部
33
3
指定手段
136
スピーカ
120
CPU
スケジュー
リング部
39
21
見積手段
6
区分手段
ディスプレイ
134
122
メモリ
判定部
126
シミュレー
ション結果
データベース
15
HDD
132

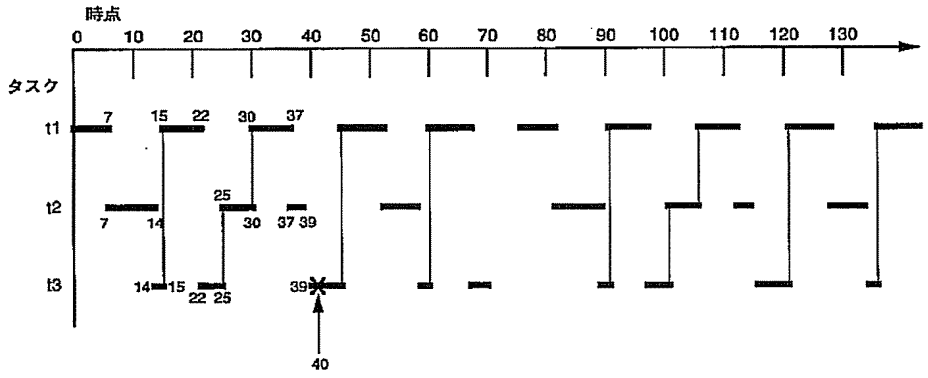
【図7】


START
1
STEP720
主スケジューラの
タスクキューが空？
YES
2
NO
STEP725
優先順位が最高の
タスクを選択
STEP730
CPUで実行
STEP735
新しいタスクは
到着したか？
YES
NO
STEP740
現在のタスクは
終了したか？
YES
NO
STEP745
高い優先順位の
時間が終了した？
NO
YES
STEP750
実行中タスクを
サブスケジューラに移す
NO
STEP785
新しいタスクが
到着したか？
YES
2
STEP752
サブスケジューラの
タスクキューが空？
YES
NO
STEP
755
次のクオンタで効用を最大
にするタスクを選択
STEP760
CPUで実行
STEP765
新しいタスクは
到着したか？
YES
1
NO
STEP470
STEP770
現在のタスクは
終了したか？
YES
2
NO
STEP775
移されてきた
タスクがある？
YES
NO
STEP780
クオンタにまだ
残りがあるか？
NO
YES

【図8】


(a)
変化量(825)
810
変化量(820)
τ"1
δ
(b)
変化量(845)
830
変化量(840)
τ"2
δ
δ

【図9】


時点
0 10 20 30 40 50 60 70 80 90 100 110 120 130
タスク
t1
t2
t3
7 15 22 30 37
7 14 25 30 37 39
14 15 22 25 39
40